The CMUnited-99 Small Robot Team

CMUnited99

Manuela Veloso, Michael Bowling, Sorin Achim

MV, MB, SA: Carnegie Mellon University

Abstract. *This paper describes the CMUnited-99 small-size robot team. The team builds on our previous RoboCup champion teams ('97 and '98). The team reuses much of the hardware, perception, and strategy software from CMUnited-98. This description summarizes some of the features from past teams that are used in CMUnited-99. It focuses on our robust motion control algorithm and the team's individual and team behaviors. We conclude with the extensions planned for CMUnited-99.*

1 Introduction

The CMUnited-99 small-size robot team is a complete, autonomous architecture composed of the physical robotic agents, a global vision processing camera over-looking the playing field, and several clients as the minds of the small-size robot players. The team builds on our previous success as two-time ('97 and '98) RoboCup champions of the small-size league.

The system uses the robots built for CMUnited-98. These robots use a differential drive mechanism for motion. The robots are controlled via an offboard computer with motion commands sent via radio communication. The robots are also equipped with a kicking device, which can be activated through radio commands.

The perception, control, and strategy software is also resued from CMUnited-98, with some additional advancements. This team description highlights the features from our past team that are being used in CMUnited-99, as well as describing the extensions planned for the team. Section 2 describes the low-level motion control algorithms that form the basis for strategic decisions. Section 3 describes the features of the high-level strategic reasoning. Section 4 describes our planned improvements.

For a detailed description of the hardware and vision system, see [3]. This also contains further details on the algorithms discussed in this description.

2 Motion Control

The goal of our low level motion control is to be as fast as possible while remaining accurate and reliable. This is challenging due to the lack of feedback from the motors, forcing all control to be done using only visual feedback. Our motion control algorithm is robust. It addresses stationary and moving targets with integrated obstacle avoidance. The algorithm makes effective use of the prediction of the ball's trajectory provided by the Kalman-Bucy filter.

We achieve this motion control functionality by a reactive control mechanism that directs a differential drive robot to a target configuration. Though based on the CMUnited-97's motion control [4], CMUnited-98/99 includes a number of major improvements. The target configuration for the motion planner has been extended. The target configuration includes: (i) the *Cartesian position*; and (ii) the *direction* that the robot is required to be facing when arriving at the target position. Obstacle avoidance is integrated into this controller. Also, the target configuration can be given as a function of time to allow for the controller to reason about intercepting the trajectory of a moving target.

2.1 Differential Drive Control for Position and Direction

CMUnited-98's basic control rules were improved from those used in CMUnited-97. The rules are a set of reactive equations for deriving the left and right wheel velocities, v_l and v_r, in order to reach a target position, (x^*, y^*):

$$\begin{aligned} \Delta &= \theta - \phi \qquad (1) \\ (t, r) &= (\cos^2 \Delta \cdot \operatorname{sgn}(\cos \Delta), \sin^2 \Delta \cdot \operatorname{sgn}(\sin \Delta)) \\ v_l &= v(t - r) \\ v_r &= v(t + r), \end{aligned}$$

where θ is the direction of the target point (x^*, y^*), ϕ is the robot's orientation, and v is the desired speed (see Figure 1(a)).[1]

We extend these equations for target configurations of the form (x^*, y^*, ϕ^*), where the goal is for the robot to reach the specified target point (x^*, y^*) while facing the direction ϕ^*. This is achieved with the following adjustment:

$$\theta' = \theta + \min\left(\alpha, \tan^{-1}\left(\frac{c}{d}\right)\right),$$

where θ' is the new target direction, α is the difference between our angle to the target point and ϕ^*, d is the distance to the target point, and c is a clearance parameter (see Figure 1(a).) This will keep the robot a distance c from the target point while it is circling to line up with the target direction, ϕ^*. This new target direction, θ', is now substituted into equation 1 to derive wheel velocities.

In addition to our motion controller computing the desired wheel velocities, it also returns an estimate of the time to reach the target configuration,

[1] All angles refer to a fixed coordinate system.

$\hat{T}(x^*, y^*, \phi^*)$. This estimate is a crucial component in our robot's strategy. It is used both in high-level decision making, and for low-level ball interception, which is described later in this section. Currently, $\hat{T}(x^*, y^*, \phi^*)$ is computed using a hand-tuned linear function of d, α, and Δ.

2.2 Obstacle Avoidance

Obstacle avoidance was also integrated into the motion control. This is done by adjusting the target direction of the robot based on any immediate obstacles in its path. This adjustment can be seen in Figure 1(b).

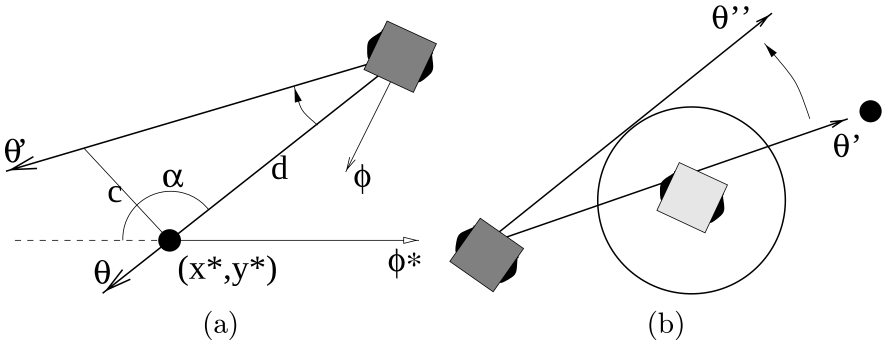

Figure 1: (a) The adjustment of θ to θ' to reach a target configuration of the form (x^*, y^*, ϕ^*); (b) The adjustment to avoid immediate obstacles.

If a target direction passes too close to an obstacle, the direction is adjusted to run tangent to the a preset allowed clearance for obstacles. Since the motion control mechanism is running continuously, the obstacle analysis is constantly replanning obstacle-free paths. This continuous replanning allows for the robot to handle the highly dynamic environment and immediately take advantage of short lived opportunities.

2.3 Moving Targets

One of the real challenges in robotic soccer is to be able to control the robots to intercept a moving ball. This capability is essential for a high-level ball passing behavior. It is achieved as an extension of the control algorithm to aim at a stationary target. Our extension allows for the target configuration to be given as a function of time, where $t = 0$ corresponds to the present,

$$f(t) = (x^*, y^*, \phi^*).$$

At some point in the future, t_0, we can compute the target configuration, $f(t_0)$. We can also use our control rules for a stationary point to find the wheel velocities and estimated time to reach this hypothetical target as if it were stationary. The time estimate to reach the target then informs us whether it is possible to reach it within the allotted time. Our goal is to find the nearest point in the future where the target can be reached. Formally, we want to find,

$$t^* = \min\{t > 0 : \hat{T}(f(t)) \leq t\}.$$

After finding t^*, we can use our stationary control rules to reach $f(t^*)$. In addition we scale the robot speed so to cross the target point at exactly t^*.

Unfortunately, t^* cannot be easily computed within a reasonable time frame. We approximate the value t^* by discretizing time with a small time step. The algorithm finds the closest of these discretized time points that satisfies our estimate constraint. The target configuration as a function of time is computed using the ball's predicted trajectory. Our control algorithm for stationary points is then used to find a path and time estimates for each discretized point along this trajectory, and the appropriate target point is selected.

3 Strategy

The main focus of our research is on developing algorithms for collaboration between agents in a team. An agent, as a member of the team, needs to be capable of individual autonomous decisions while, at the same time, its decisions must contribute towards the team goals.

CMUnited-97 introduced a flexible team architecture in which agents are organized in *formations* and *units*. Each agent plays a *role* in a unit and in a formation [2, 4]. CMUnited-98/99 builds upon this team architecture by defining a set of roles for the agents. It also introduces improvements within this architecture to help address the highly dynamic environment. CMUnited-98/99 uses the following roles: goalkeeper, defender, and attacker.

3.1 Goalkeeper

The ideal goalie behavior is to reach the expected entry point of the ball in the goal *before* the ball reaches it. Assuming that the prediction of the ball trajectory is correct and the robot has a uniform movement, we can state the ideal goalie behavior. Let v_g and v_b be the velocities of the goalie and of the ball respectively, and d_g and d_b be the distances from the goalie and the ball to the predicted entry point. We want $\frac{d_g}{v_g} = \frac{d_b}{v_b} - \epsilon$, where ϵ is a small positive value to account for the goalie reaching the entry point slightly before the ball.

Unfortunately, the ball easily changes velocity and the movement of the robot is nonuniform and uncertain. Therefore we have followed a switching behavior for the goalie based on a threshold of the ball's estimated trajectory. If the ball's estimated speed is higher than a preset threshold, the goalie moves directly to the ball's predicted entry goal point. Otherwise, the goalie selects the position that minimizes the largest portion of unobstructed goal area, by finding the location that bisects the angles of the ball and the goal posts.

3.2 Defender

A defender's purpose is two-fold:

1. to stop the opponents from scoring in our goal; and
2. to not endanger our own goal.

The first goal is clearly a defender's role. The second goal comes as the result of the uncertain ball handling by the robots. The robots can easily push (or touch) the ball unexpectedly in the wrong direction when performing a difficult maneuver.

To achieve the two goals, we implemented three behaviors for the defender. *Blocking*, illustrated in Figure 2(a), is similar to the goalie's behavior except that the defender positions itself further away from the goal line. *Clearing*, illustrated in Figure 2(b), pushes the ball out of the defending area. It does this by finding the largest angular direction free of obstacles (opponents and teammates) that the robot can push the ball towards. *Annoying*, illustrated in Figure 2(c), is somewhat similar to the goal-keeping behavior except that the robot tries to position itself between the ball and the opponent nearest to it. This is an effort to keep the opponent from reaching the ball.

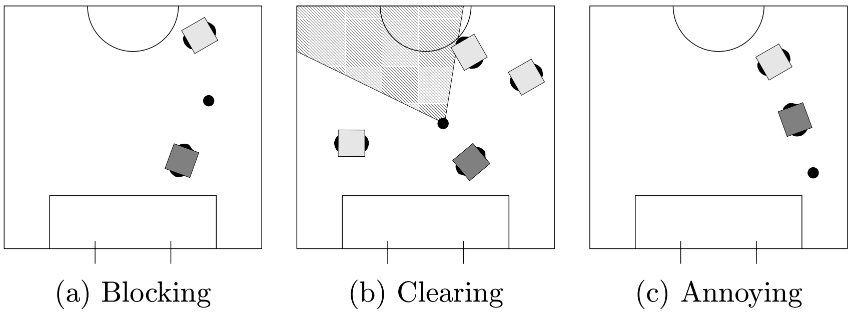
(a) Blocking (b) Clearing (c) Annoying

Figure 2: The defender's behaviors. The dark and light robots represent the defender and the opponents respectively.

Selecting when each of these behaviors is used is very important to the effectiveness of the defender. For example, clearing the ball when it is close to our own goal or when it can bounce back off another robot, can lead to scoring in our own goal. We used the decision tree in Figure 3 to select which action to perform based on the current state.

The two attributes in the tree, namely *Ball Upfield* and *Safe to Clear*, are binary. *Ball Upfield* tests whether the ball is upfield (towards the opponent's goal) of the defender. *Safe to Clear* tests whether the open area is larger than a preset angle threshold. If *Ball Upfield* is true, the defender clears or blocks, depending on the value of *Safe to Clear*. If *Ball Upfield* is false then the ball is closer to the goal than the defender and the robot *annoys* the attacking robot.

3.3 Attackers

Attacking involves one of the best opportunities for collaboration, and much of the innovation of CMUnited-98/99 has been developing techniques for finding and exploiting these opportunities.

In many multi-agent systems, one or a few agents are assigned, or assign themselves, the specific task to be solved at a particular moment. We

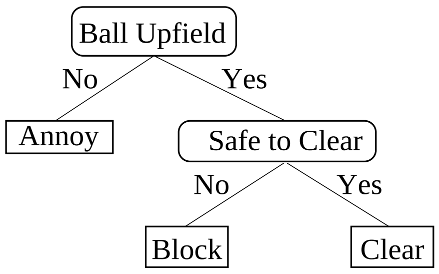

Figure 3: The decision tree for the defender's behavior.

view these agents as the *active* agents. Other team members are *passive* waiting to be needed to achieve another task or assist the active agent(s). This simplistic distinction between active and passive agents to capture teamwork was realized in CMUnited-97. The agent that goes to the ball is viewed as the active agent, while the other teammates are passive.

CMUnited-98/99 significantly extends this simplistic view in two ways: (i) we use a decision theoretic algorithm to select the active agent; and (ii) we use a technique for passive agents *to anticipate* future collaboration. Passive agents are therefore not actually "passive;" instead, they actively *anticipate* opportunities for collaboration. This collaboration relies on robust individual behaviors.

3.3.1 Individual Behaviors

We first developed individual behaviors for passing and shooting. Passing and shooting in CMUnited-98/99 is handled effectively by the motion controller. The target configuration is specified to be the ball (using its estimated trajectory) and the target direction is either towards the goal or another teammate. This gives us robust and accurate individual behaviors that can handle obstacles as well as intercepting a moving ball.

3.3.2 Decision Theoretic Action Selection

Given the individual behaviors, we must select an active agent and appropriate behavior. This is done by a decision theoretic analysis using a single step look-ahead. With n agents this amounts to n^2 choices of actions involving shooting or a pass to another agent followed by that agent shooting. An estimated probability of success for each pass and shot is computed along with the time estimate to complete the action, which is provided by the motion controller. A value for each action is computed,

$$\text{Value} = \frac{\text{Pr}_{\text{pass}}\text{Pr}_{\text{shoot}}}{\text{time}}.$$

The action with the largest value is selected, which determines both the active agent and its behavior. Table 1 illustrates an example of the values for the selection considering two attackers, 1 and 2.

It is important to note that this action selection is occurring on each iteration of control, i.e., approximately 30 times per second. The probabilities of success, estimates of time, and values of actions, are being continuously

		Probability of Success			
Attacker	Action	Pass	Shoot	Time(s)	Value
1	Shoot	–	60%	2.0	0.30
1*	Pass to 2	60%	90%	1.0	0.54
2	Shoot	–	80%	1.5	0.53
2	Pass to 1	50%	40%	0.8	0.25

Table 1: Action choices and computed values are based on the probability of success and estimate of time. The largest-valued action (marked with the *) is selected.

recomputed. This allows for quick changes of actions if shooting opportunities become available or collaboration with another agent appears more useful.

3.3.3 Dynamic Positioning (SPAR)

Although there is a clear action to be taken by the active agent, it is unclear what the passive agents should be doing. For CMUnited-98/99, we introduce a team-based notion of *anticipation.* The passive team members position themselves strategically so as to optimize the chances that their teammates can successfully collaborate with them, in particular pass to them.

This strategic position takes into account the position of the other robots (teammates and opponents), the ball, and the opponent's goal. The position is found using an algorithm, which we call SPAR for *Strategic Positioning with Attraction and Repulsion.* The selected position is the solution to a multiple-objective function with repulsion and attraction points. This extends similar approaches using potential fields [1], to our highly dynamic, multi-agent domain.

The usefulness of collaboration is directly related to how "open" a position is to allow for a successful pass. It is also directly related to how likely that pass would result in a goal. SPAR approximates this measure by maximizing the repulsion from other robots and minimizing attraction to the ball and to the goal, namely:

- *Repulsion* from opponents, O_i: $\forall i, \max dist(P, O_i)$.
- *Repulsion* from teammates. T_i: $\forall i, \max dist(P, T_i)$.
- *Attraction* to the ball, B: $\min dist(P, B)$.
- *Attraction* to the opponent's goal, G: $\min dist(P, G)$.

This is a multiple-objective function. To solve this optimization problem, we restate this function into a single-objective function. As each term in the multiple-objective function may have a different relevance (e.g., staying close to the goal may be more important than staying away from opponents), we want to consider different functions of each term. We weight the terms differently, namely w_{O_i}, w_{T_i}, w_B, and w_G, for the weights for opponents, teammates, the ball, and the goal, respectively. This gives us a weighted single-objective function:

$$\max \left(\begin{array}{l} \sum_{i=1}^{n} w_{O_i}\, dist(P, O_i) + \sum_{i=1}^{n} w_{T_i}\, dist(P, T_i) - \\ -w_B\, dist(P, B) - w_G\, dist(P, G) \end{array} \right)$$

This optimization problem is then solved under the following set of constraints:

- Do not block a direct shot from the active teammate.
- Do not stand behind other robots, because these are difficult positions to receive passes from the active teammate.

The solution to this optimization problem under constraints gives us a target location for the "passive" agent. Figure 4 (a) and (b) illustrate these two constraints. Figure 4 (c) shows the combination of these two sets of constraints and the resulting position returned by our algorithm for the anticipating passive teammate.

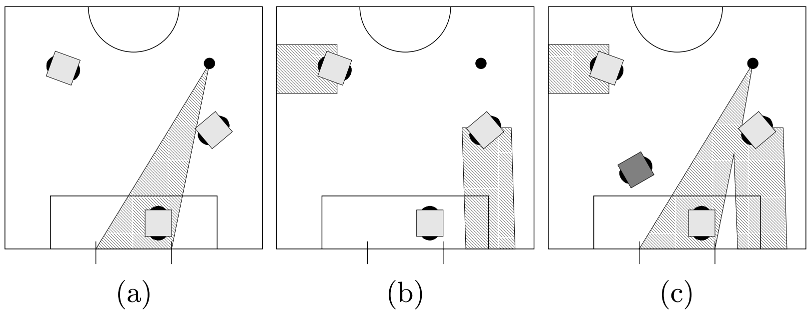

Figure 4: Constraints used in SPAR and the selected position returned by the algorithm. Shaded regions show areas eliminated by the constraints; (a) do not block goal shot, and (b) avoid difficult collaboration. The combined constraints and selected position using SPAR is shown in (c).

4 Extensions

CMUnited-99 will include a number of advancements from CMUnited-98. These advancements include:

- Physical rebuilding of the goalkeeper robot to conform to the RoboCup '99 size constraints, while retaining a broad surface to defend the goal.
- Extension of the motion control algorithm to allow for a preferred side, so that attackers will make more use of the kicking device.
- Improve the coordination between the goalkeeper and the defender. This will also allow for multiple defenders to coordinate their actions.
- Additional strategic planning to dynamically adjust the number of attackers and defenders based on the current situation (e.g. position of the ball, the "formation" of the opponent, or current score).

CMUnited-99 will compete in RoboCup-99 in Stockholm.

References

[1] Jean-Claude Latombe. *Robot Motion Planning.* Kluwer Academic Publishers, Boston, 1991.

[2] Peter Stone and Manuela Veloso. The CMUnited-97 simulator team. In Hiroaki Kitano, editor, *RoboCup-97: Robot Soccer World Cup I.* Springer Verlag, Berlin, 1998.

[3] Manuela Veloso, Michael Bowling, Sorin Achim, Kwun Han, and Peter Stone. The CMUnited-98 champion small robot team. In Minoru Asada and Hiroaki Kitano, editors, *RoboCup-98: Robot Soccer World Cup II.* Springer Verlag, Berlin, 1999.

[4] Manuela Veloso, Peter Stone, and Kwun Han. CMUnited-97: RoboCup-97 small-robot world champion team. *AI Magazine*, 19(3):61–69, 1998.